TD Informatique TSI2

Chap 5. Machine_Learning_Part1

EXERCICE 01

Trier des points. Soit un ensemble de points dans l'espace muni d'un repère orthonormé d'origine O. Un point possède des coordonnées reprsentées par une liste du type `[x,y,z]`. L'objectif est de trier ces points en fonction de leur distance (au carré) à un point donné P, de la plus petite à la plus grande.

1. Écrire une fonction `distance` qui prend en argument deux listes représentant les coordonnées de deux points quelconques et renvoie le carré de la distance euclidienne entre ces deux points.
2. Écrire une fonction `compare` qui prend en paramètres trois listes `p,p1,p2` représentant dans l'ordre le point P et deux points quelconques P_1, P_2 et qui renvoie -1 si P_1 est plus proche de P que P_2, 1 si P_2 est plus proche de P que P_1 et 0 si les deux points sont équidistants de P.
   On pose :

```
liste = [[1, 1, -1], [2, 0, 0], [1, -1, 1], [2, 1, -1]]
P = [0,0,0]
```

   Calculer `compare(P,liste[i],liste[j])` pour tout i et j possibles avec $i < j$.
3. On considère la procédure suivante qui a pour argument un point `p` et une liste nommée `new_liste` composée de listes de trois points, qui applique compare à `p` et à deux points quelconques de `new_liste` et qui renvoie la liste `new_liste` avec les points triés par ordre croissant des distances entre ces points et le point `p`.

```
def tri_points(p,new_liste):
    for i in range(len(new_liste)-1) :
        i_mini = i
        mini = new_liste[i]
        for j in range(i+1, len(new_liste)):
            if compare(p,mini, new_liste[j]) == 1:
                i_mini = j
                mini = new_liste[j]
        new_liste[i],new_liste[i_mini] = new_liste[i_mini],new_liste[i]
    return new_liste
```

   L'appliquer à `tri_points(P,liste)` de la question **2.** Retrouver le résultat avec ce que vous avez trouvé à la question **2.**

EXERCICE 02

1. Taper les instructions suivantes :

```
In [1]: example_ens=[1,6,3,4,5,0,2,-2,-7,45]
In [2]: d={elt:False for elt in example_ens}
In [3]: d
In [4]: d[example_ens[3]]="BlaBla"
In [5]: d
In [6]: from random import randrange
In [7]: i=randrange(4,8)
In [8]: i
In [9]: d[example_ens[i]]=2
In[10]: d
```

T.S.V.P →

Quel est le type de `d` ? Expliquer les différentes transformations de `d`

2. On considère la procédure suivante.

```
In [14]: def distribute_incomplete(ens,k):
    ...:     n = len(ens)
    ...:     d = {elt: False for elt in ens}
    ...:     d[ens[0]] = 1
    ...:     for p in range(1,k):
    ...:         i = randrange(p,n)
    ...:         d[ens[i]] = p+1
    ...:         ens[i],ens[p] = ens[p],ens[i]
    ...:     return d
    ...:
```

Puis on suppose que l'on tape aussi ce qui suit.

```
In [10]: example_ens=[1,6,3,4,5,0,2,-2,-7,45]
In [11]: distribute_incomplete(example_ens,4)
# ON OBTIENT ALORS :
Out[16]:
{1: 1,
 6: False,
 3: 3,
 4: False,
 5: 2,
 0: False,
 2: 4,
 -2: False,
 -7: False,
 45: False}
```

Expliquer à la main ce que la machine a fait quasiment spontanément !
Vérifier en particulier que le nouveau `example_ens` est bien :

```
In [12]: example_ens
Out[12]: [1, 5, 3, 2, 6, 0, 4, -2, -7, 45]
```

Taper alors tout ce code Python vous aussi. Attention, comme `randrange` est une fonction booléenne, vous obtiendrez certainement un autre `d` et un autre `example_ens` que ceux affichés ici.

Pour la suite, reprendre `example_ens` du début, c'est-à-dire :

```
In [10]: example_ens=[1,6,3,4,5,0,2,-2,-7,45]
```

3. Écrire alors une fonction `distribute` qui prend en paramètres un ensemble donné `ens` sous la forme d'une liste et un nombre entier `k` non nul qui reprend `distribute_incomplete` en y ajoutant une boucle `for elt in ens[k:n]:` dans laquelle on mettra `d[elt] = randrange(1,k+1)` qui règle le problème des eléments de `d` restés `False`

   Tapez ensuite `distribute(example_ens,2)` et `distribute(example_ens,5)`.
   Que fait donc `distribute` ?

4. Écrire une fonction `unterteilen` qui prend en arguments un ensemble donné `ens` sous la forme d'une liste et un entier non nul `k` et qui renvoie la partition (nommée `parties` dans la procédure) en `k` parties de l'ensemble `ens` (contenant au moins `k` éléments) générée aléatoirement à l'aide de la fonction `distribute`.

   Indication : on commencera par affecter à la variable `d` le dictionnaire `distribute(ens, k)`.
   Puis on rentre dans `parties` une liste remplie de `k` fois la liste vide `[]`.
   Puis on fera une boucle `for elt in d:` et dans cette boucle, on affectera chaque `elt` dans la liste de `parties` qui est en position `d[elt] - 1`.

5. Appliquer `unterteilen` à `examples_ens` avec `k=3`, `k=6` et `k=1`.